**Dự án Adam Store**

# **I ) Các thông tin trong dự án :**

**- Tài khoản Admin mặc định :**

**Email :** [**admin@gmail.com**](mailto:admin@gmail.com)

**Password : 123456**

**- Phối màu** (tham khảo) : <https://colorhunt.co/>

1. **Các entity trong dự án:**

* **User** (người dùng) : id, name, email, password, avatarUrl, age, gender, status (enum : ACTIVE|INACTIVE), createdBy, updatedBy, createdAt, updatedAt , (1-1) **Cart**, (n-n) **Role**, (1-n) **Order**, (1-n) **Address**, (1-n) **Review,**  (1-n) **PromotionUsage**
* **Category** (danh mục sản phẩm) : id, name, description, status (enum : ACTIVE|INACTIVE), createdBy, updatedBy, createdAt, updatedAt (1-n) **Product**
* **Product** (sản phẩm ) : id, name, description, isAvailable ( hàng còn thì sẽ là true), averageRating, soldQuantity, totalReviews, status (enum : ACTIVE|INACTIVE), createdBy, updatedBy, createdAt, updatedAt **|**  (n-1) **Category,**  (1-n) **Review,** (1-n) **ProductImage ,** (1-n) **ProductVariant**
* **ProductImage :** id , publicId, fileName, imageUrl, (n-1) **Product**
* **ProductVariant** (biến thể sản phẩm)**:** id, price, quantity(số lượng hàng còn trong kho), isAvailable ( hàng còn thì sẽ là true), (n-1) **Product**, (n-1) **Color**, (n-1) **Size**, (1-n) **OrderItem,** (1-n) **OrderItem**
* **Color** ( màu) : id, name, (1-n) **ProductVariant**
* **Size** : id, name, (1-n) **ProductVariant**
* **PaymentHistory** ( lịch sử thanh toán của đơn hàng ) : id, isPrimary (khi đơn hàng hoàn tất chuyển thành true), paymentMethod, totalAmount, paymentStatus (enum : PAID | PENDING | REFUNDED | CANCELED | FAILED ), paymentTime, (n-1) **Order**
* **Order** ( đơn hàng ) :id, orderDate, totalPrice, orderStatus (enum : PENDING |PROCESSING | SHIPPED | DELIVERED| CANCELLED), createdAt, updatedAt (1-n) **OrderItem**, (n-1) **User**, (1-n) **PaymentHistory,** (n-1) **Address,**  (1-1) **PromotionUsage**
* **OrderItem** ( chi tiết từng đơn hàng ) : id, quantity, unit\_price, (n-1) **ProductVariant** , (n-1) **Order**
* **CartItem** ( Sản phẩm trong giỏ hàng ) : id, price, quantity(số lượng sp trong giỏ hàng), (n-1) **ProductVariant**, (n-1) **Cart**
* **Cart** (Giỏ hàng) : id, createdAt, updatedAt (1-1) **User,** (1-n) **CartItem**
* **Review** ( đánh giá sản phẩm ) : id, rating, comment, imageUrl, createdAt, updatedAt | (n-1) **User**, (n-1) **Product**
* **Ward** (Phường/Xã) : code, name, (n-1) **District**
* **District** ( Quận / Huyện / TP thuộc tỉnh ) : id, name, (n-1) **Province**
* **Province** ( tỉnh ) : id, name
* **Address** ( địa chỉ gồm Số nhà - Quận/Huyện/Thành phố - Tỉnh/Thành phố) : id, isDefault, isVisible, phone, streetDetail, (n-1) **Ward**, (n-1) **District**, (n-1) **Province**, (n-1) **User,** (1-n) **Order**
* **Promotion** ( khuyến mãi ) : id, title, description, discount\_percent, start\_date, end\_date, status (enum : ACTIVE|INACTIVE), createdBy, updatedBy, createdAt, updatedAt | (1-n) **PromotionUsage**
* **Branch** (chi nhánh cửa hàng) : id name, location, phone, status (enum : ACTIVE|INACTIVE), createdBy, updatedBy, createdAt, updatedAt
* **Role** (vai trò) : id, name, description, (n-n) **User**, (n-n) **Permission**
* **Permission** (quyền hạn): id, name, module, apiPath, method (n-n) **Role**
* **PromotionUsage** (thông tin áp dụng khuyến mãi): id, discountAmount, usedAt, ( n-1 ) **User**, (n-1) **Promotion**, (1-1) **Order**

- Các trạng thái orderStatus :

PENDING : đang chờ

PROCESSING : Đang xử lý

SHIPPED : Đã giao hàng

DELIVERED : Đã giao thành công

CANCELLED : Đã hủy

- Các trạng thái PaymentHistory (paymentStatus) :

PAID : Giao dịch đã thanh toán thành công

PENDING : Giao dịch đang chờ xử lý, chưa xác nhận thành công

REFUNDED : Giao dịch đã được hoàn tiền cho khách hàng

CANCELED : Giao dịch bị hủy trước khi hoàn tất

FAILED : Thanh toán thất bại do lỗi kỹ thuật, từ chối ngân hàng,...

1. **Các đối tượng được Admin quản lý ( các trang quản lý của Admin ) :**

* Quản lý Branch( thêm, sửa, xóa)
* Quản lý Category (thêm, sửa, xóa)
* Quản lý Promotion (thêm, sửa, xóa)
* Quản lý Color (thêm, sửa, xóa)
* Quản lý Size (xem)
* Quản lý Ward (xem)
* Quản lý District (xem)
* Quản lý Province (xem)
* Quản lý Address (xem, xóa)
* Quản lý Role (xem)
* Quản lý permission (xem)
* Quản lý Order (xem)
* Quản lý Revenue (doanh thu) (xem) : quản lý theo khoảng thời gian từ startDate đến endDate. Tạo biểu đồ cho doanh thu các tháng trong khoảng thời gian. Hiển thị thêm thông tin các đơn hàng được đặt trong khoảng thời gian đó
* Quản lý Payment History (xem) : lọc theo trang, theo ngày, theo giờ  
  - Quản lý Product (thêm, sửa, xóa) : thêm Product sẽ đi kèm với tạo các Product Variant
* Quản lý Product Variant : giúp quản lý theo từng Product ( Product sẽ có những Product Variant nào), sẽ xem được quantity và price của từng Product Variant

### 3.Xóa các thực thể :

**a) Xóa mềm** (xóa mềm các đối tượng nên dữ liệu các đối tượng vẫn còn trong hệ thống, chuyển status từ ACTIVE => INACTIVE, có thể khôi phục => có thêm Api Restore để khôi phục) : **User, Category, Branch, Promotion, Product, Product Variant, Address**

- **User** : Chỉ xóa mềm khi không có đơn hàng nào đang trong trạng thái hoạt động (PROCESSING, SHIPPED, DELIVERED) và user chưa sử dụng một promotion nào cả (cả ACTIVE và INACTIVE) (user phải chưa sử dụng dịch vụ nào bên web)

=> Sẽ xuất hiện 2 API :

* Fetch All User for Admin (gồm cả User có status ACTIVE và INACTIVE) để admin quản lý
* Restore User : dùng để khôi phục các User đã bị soft delete

- **Branch** : không ảnh hưởng

=> Sẽ xuất hiện 3 API :

* Fetch All Branch for Admin (gồm cả Branch có status ACTIVE và INACTIVE) để admin quản lý
* Fetch All Branch for User (chỉ có Branch có status ACTIVE) : để user có thể thấy các Branch còn hoạt động
* Restore Branch: dùng để khôi phục các Branch đã bị soft delete

- **Category** : chỉ được xóa mềm nếu không có bất kỳ sản phẩm (Product) nào có status = ACTIVE (đang hoạt động) thuộc danh mục đó. Nếu vẫn còn sản phẩm đang hoạt động, không được phép xóa mềm danh mục

=> Sẽ xuất hiện 2 api :

* Fetch All Category for Admin (gồm cả Category có status ACTIVE và INACTIVE) để admin quản lý
* Fetch All Category for User (chỉ có Category có status ACTIVE) : để user có thể thấy các Category còn hoạt động
* Restore Category : dùng để khôi phục các Category đã bị soft delete

- **Promotion**: Được phép xóa mềm nếu không còn bất kỳ đơn hàng nào sử dụng chương trình khuyến mãi này

=> Có 3 API :

* Fetch All Promotion for Admin (gồm cả Promotion có status ACTIVE và INACTIVE) để admin quản lý
* Fetch All Promotions Available By User (chỉ có Promotion có status ACTIVE) : đối với từng User chỉ có thể lấy và sử dụng các Promotion đang hoạt động (status = ACTIVE) và chưa bị chính User đó sử dụng
* Restore Promotion : dùng để khôi phục các Promotion đã bị soft delete

- **Product**: Sản phẩm có thể xóa mềm nếu không có đơn hàng nào đang sử dụng sản phẩm đó (không có order nào đang sử dụng các Product Variant của Product). Nếu có đơn hàng liên quan, sản phẩm sẽ không được phép xóa mềm

=> Sẽ có 5 API :

* Fetch All Product for Admin: Lấy tất cả các sản phẩm với status ACTIVE và INACTIVE cho phép Admin quản lý
* Fetch All Product For User : Lấy tất cả các sản phẩm có status ACTIVE cho phép user xem các sản phẩm còn hoạt động
* Fetch Products By Category for User: Lấy tất cả các sản phẩm có status ACTIVE và thuộc Category cho phép user xem
* Fetch Products By Category for Admin:Lấy tất cả các sản phẩm với status ACTIVE và INACTIVE thuộc Category cho phép Admin quản lý
* Restore Product : API khôi phục các sản phẩm đã bị soft delete (kèm theo khôi phục tất cả Product Variant của Product)

- **Product Variant**: Có thể xóa mềm nếu không có đơn hàng nào liên quan đến Product Variant đó. Nếu có đơn hàng liên quan, Product Variant đó không thể xóa mềm

* Fetch All Product Variant By Product for Admin: Lấy tất cả các biến thể sản phẩm với status ACTIVE và INACTIVE theo Product để Admin quản lý
* Fetch All Product Variant By Product for User: Lấy tất cả các biến thể sản phẩm có status ACTIVE theo Product để user xem các biến thể còn hoạt động
* Restore Product Variant : API khôi phục các Product Variant đã bị soft delete
* **Address** : Có thể xóa mềm nếu chưa có order nào sử dụng address này,không được xóa address mặc định ( isDefault = true )
* Hide Address : cho user xóa chính địa chỉ của họ khỏi giao diện người dùng
* Soft Delete Address : cho admin soft delete address
* Restore Address : khôi phục địa chỉ sau khi soft delete
* Get All Address for User : lấy tất cả địa chỉ đang ACTIVE (đang hoạt động) và isVisible == true ( không bị user xóa)
* Fetch All Address for Admin : lấy tất cả address trong hệ thống ( gồm cả INACTIVE và isVisible = false )

**b) Xóa cứng** (xóa dữ liệu khỏi DB và không thể khôi phục được) : **Review, Order, PaymentHistory, Color, ProductImage**

- Xóa cứng Review : không ảnh hưởng

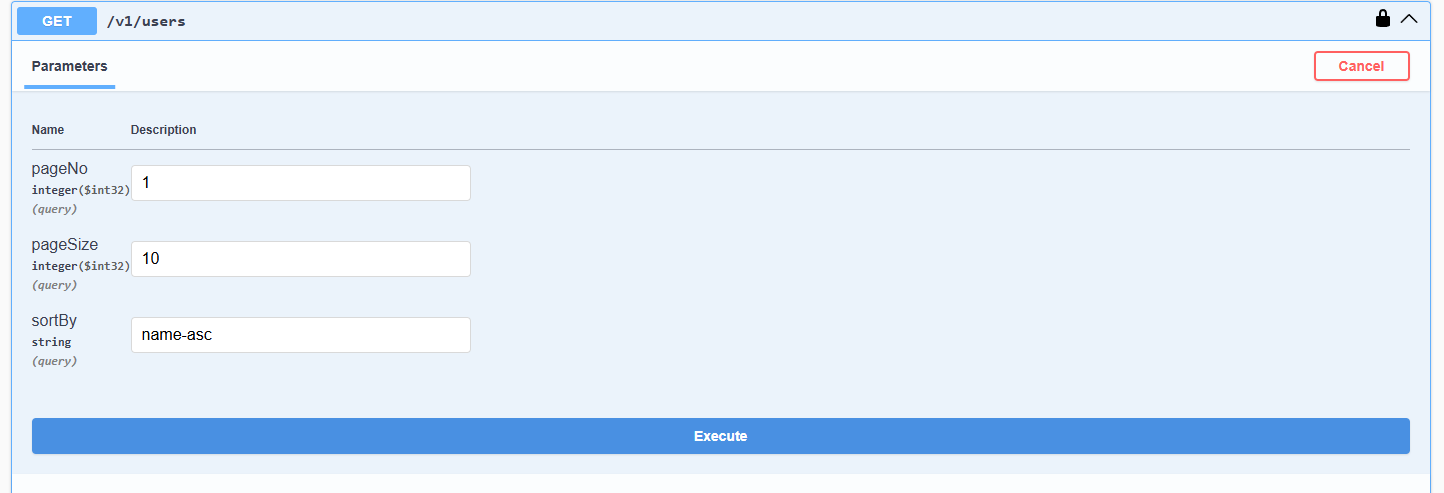
- Xóa cứng Order : không ảnh hưởng

- Xóa cứng Payment History : không ảnh hưởng

- Xóa cứng Color : chỉ cho xóa color khi chưa gắn color vào Product Variant

- Xóa cứng Product Image : không ảnh hưởng

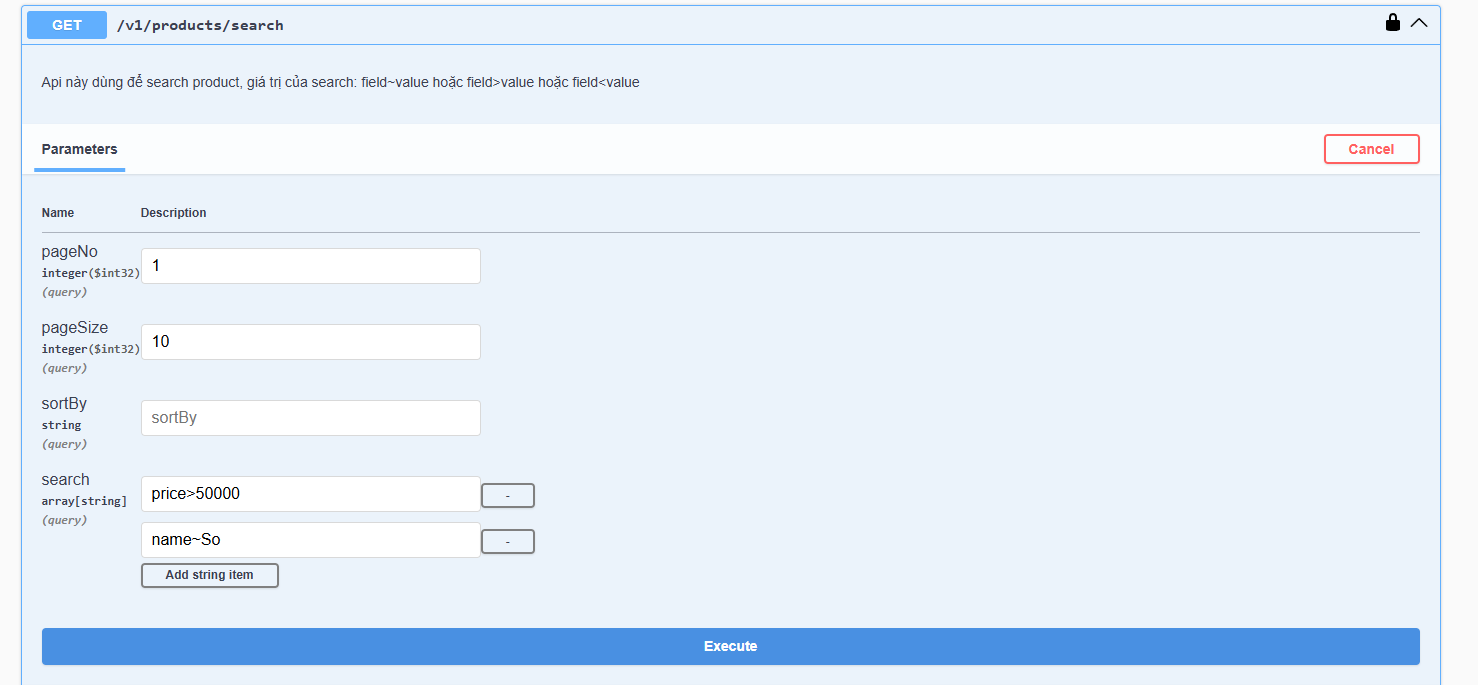
1. **Hướng dẫn sử dụng API :**

* Đối với tất cả **API fetch All** : đều có pageNo(trang muốn lấy), pageSize(số lượng phần tử mỗi trang), **sortBy**: để sắp xếp cách truyền là : **field-asc** (sắp xếp theo thuộc tính field theo thứ tự tăng dần) hoặc **field-desc** (sắp xếp theo thuộc tính field theo thứ tự giảm dần)
* 
* Đối với **Api Search Product** : thuộc tính search : các truyền vào : **field~value** hoặc **field>value** hoặc **field<value**  (riêng với Product: field có thể là **price**)

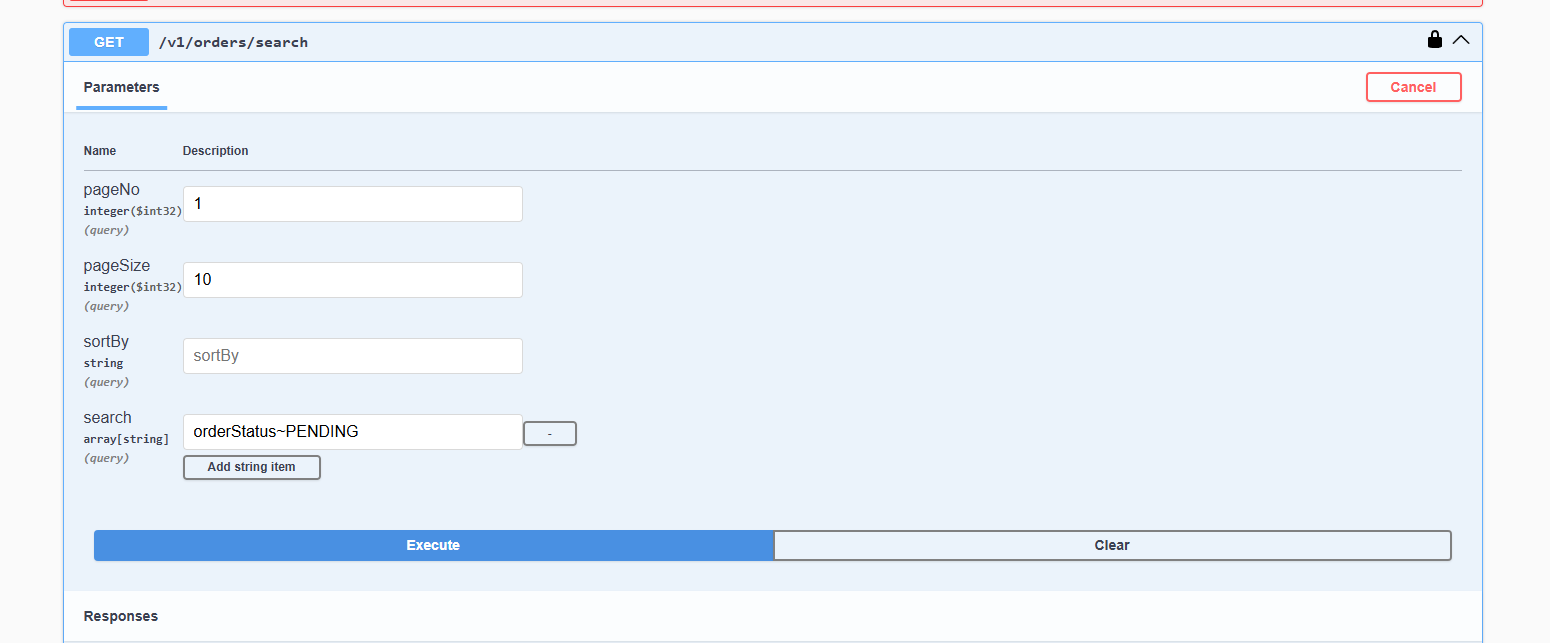
VD: **name~ao** (tìm tất cả phần tử chứa ao trong name của product), hoặc

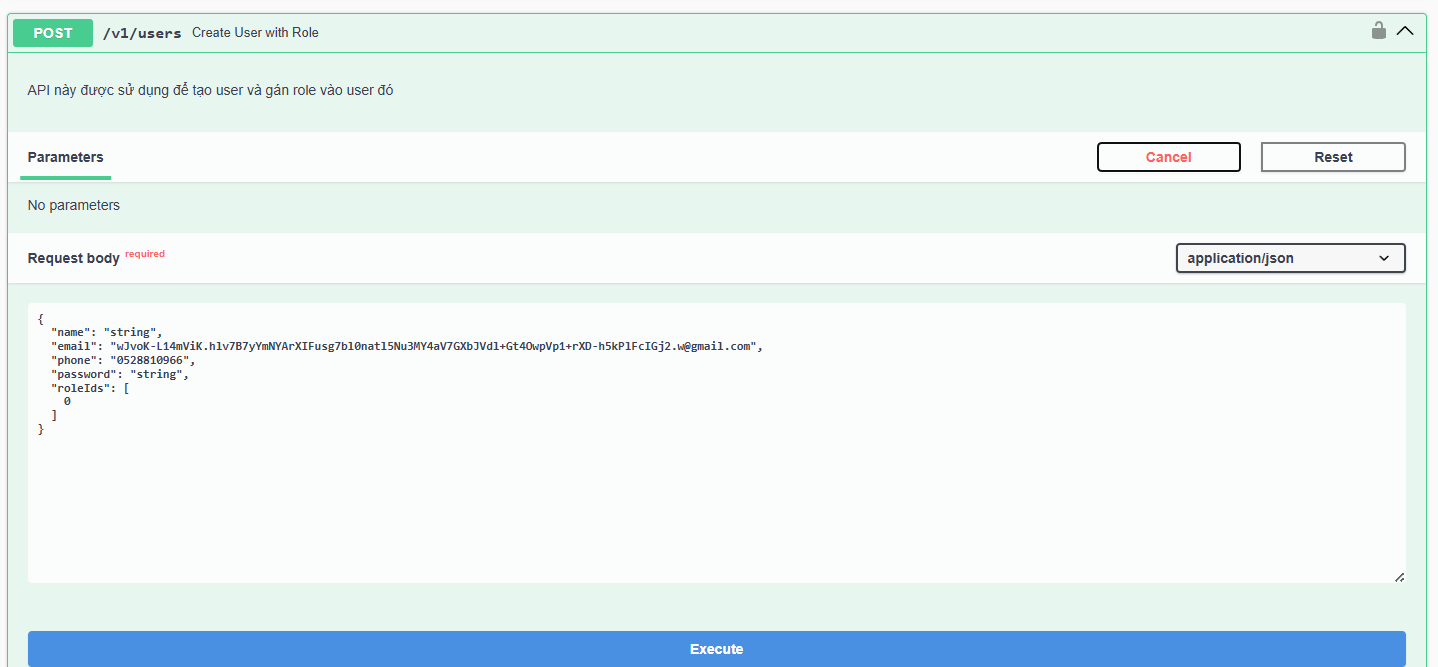
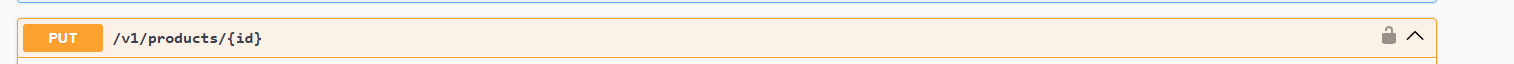
price**>500000** (tìm tất cả phần tử product có price lớn hơn **500000**) hoặc

**averageRating<4.5** (tìm tất cả phần tử product có averageRating bé hơn 4.5)

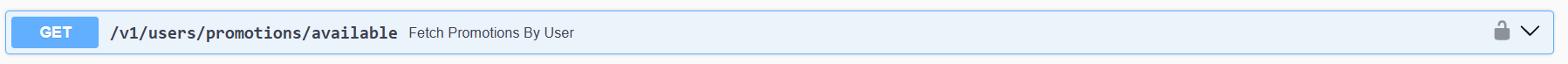


* Đối với **API search Order** (dùng cho cả user hoặc admin): có thể sử dụng với mục đích tìm kiếm, filter, lọc các order có orderStatus = PEDDING,... sử dụng thuộc tính search **field~value** hoặc **field>value** hoặc **field<value**
* Đối với user hệ thống sẽ tự động nhận diện và trả về các đơn hàng của chính user đó (thông qua token)
* Đối với admin hệ thống sẽ lấy tất cả các đơn hàng của tất cả user

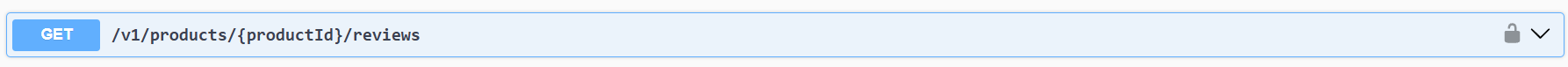


* **API thêm Admin mới**: Khi muốn thêm một Admin mới (ngoài admin mặc định là [admin@gmail.com](mailto:admin@gmail.com)) thì gọi api Create User, chỉ cần truyền RoleIds là Id của Admin vào, api này dành cho admin để có thể thêm các admin khác và user khác trong trang quản lý user
* Đối với các api Update thì với các trường không có nhu cầu update thì không cần truyền vào request. Chỉ riêng api update product luôn cần truyền 2 trường price và quantity, còn lại các API khác (của các thực thể khác) chỉ cần truyền các trường mình muốn update 

- API lấy tất cả Promotion người dùng hiện tại có thể dùng ( các promotion còn hạn và chưa được user hiện tại sử dụng ) :



- API lấy tất cả review của một Product :



- API lấy tất cả Product Variant của một Product dành cho USER xem các product variant đang còn hàng và còn hoạt động :



- API lất tất cả Product Variant của một Product cho ADMIN xem tất cả Product Variant kể cả hết hàng, còn hàng, đã xóa mềm, hay hoạt động :



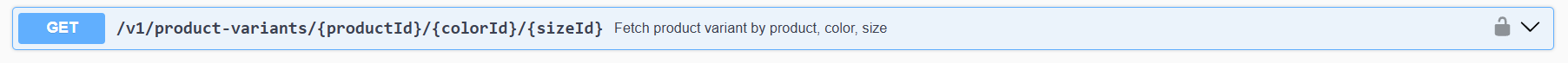
- API lấy tất cả Product đang còn hoạt động cho User xem (mục đích trang chủ) :



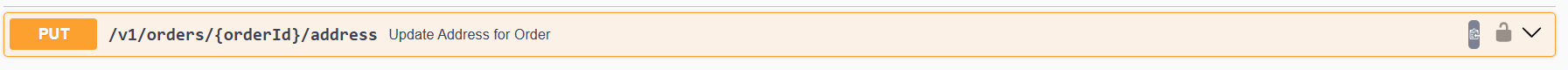
- API lấy tất cả Product trong hệ thống (cả đã xóa mềm và còn hoạt động) :



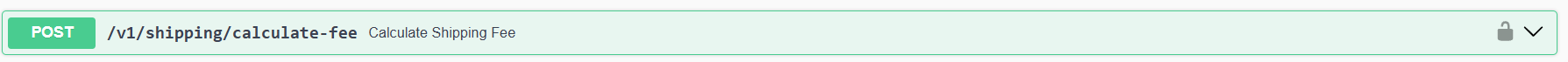
- API lấy Product Variant theo productId colorId sizeId



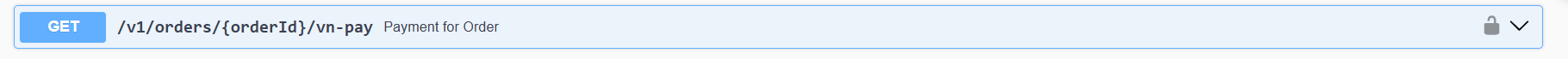
- API thay đổi địa chỉ nhận hàng khi đơn hàng trạng thái PENDING hoặc PROCESSING



- API tính phí ship :



- API thanh toán qua VNPAY (tạo URL chuyển hướng sang trang thanh toán VNPAY) :



- Sau khi thanh toán xong chuyển về FE và FE gọi API callback để update và BE nhận thông thanh toán (thành công hay thất bại) : 

- API thanh toán lại, sau khi thanh toán thất bại :



- API lấy tất cả product đang còn hoạt động theo Category để user xem khi tích vào từng Category :

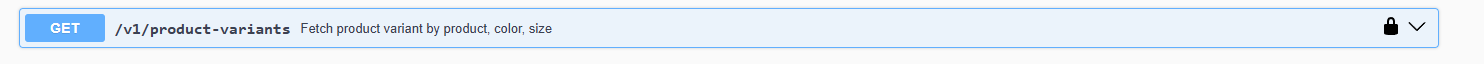


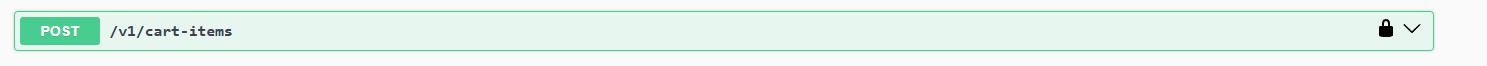
- API lấy tất cả product (đã xóa mềm và hoạt động) theo Category để Admin xem để quản lý :



# **II) Nghiệp vụ và cách thực hiện :**

1. **Quy trình thêm vào giỏ hàng :**

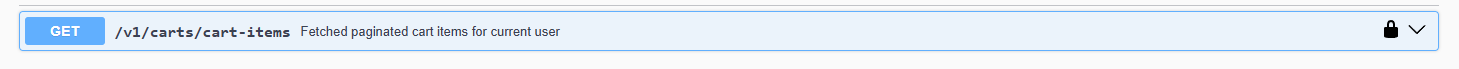
* Lấy User hiện tại từ token và truy vấn Cart tương ứng với người dùng đó (đối với mỗi user sẽ tự được tạo một cart tương ứng)
* Khi đang xem sản phẩm nên sẽ lấy được thông tin productId, size muốn chọn, color muốn chọn, gọi api để tìm ProductVariant từ đó tìm được ProductVariant
* 
* Kiểm tra ProductVariant hợp lệ (đối với mỗi size, color tương ứng check xem còn quantity(số lượng hàng trong kho còn không) )
* Gọi API tạo CartItem hệ thống sẽ kiểm tra CartItem đã tồn tại dựa vào ProductVariant đã có trong giỏ hàng chưa
* Nếu có, cập nhật số lượng quantity của CartItem
* Nếu chưa, thêm mới một CartItem vào giỏ hàng



* Đối với các sản phẩm trong giỏ hàng có thể update số lượng thông qua :



* Để lấy tất cả các sản phẩm đã thêm vào giỏ hàng :



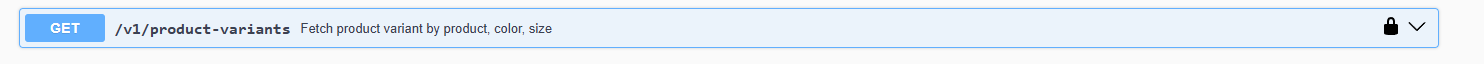
1. **Quy trình đặt hàng :**

**Đặt hàng :**

**+ Thanh toán ngay**

**+ Thêm vào giỏ hàng rồi mới thanh toán**

1. **Thanh toán ngay :**

* Khách hàng duyệt sản phẩm từ Product (thông qua bảng Product và Category)
* Chọn biến thể sản phẩm (ProductVariant) gồm màu (Color) và kích thước (Size) kèm số lượng sản phẩm muốn mua (quantity)
* Hệ thống kiểm tra số lượng tồn kho của sản phẩm. Nếu sản phẩm còn đủ số lượng, hệ thống sẽ tiếp tục (lỗi sẽ thông báo không đủ số lượng sản phẩm)
* Lúc này sẽ gọi api để lấy product variants :
* Tiến hành gọi các api để lấy thông tin về address và promotion để lấy cho phần địa chỉ nhận hàng và mã giảm giá của tôi
* Lưu ý gọi API để lấy các Promotion có sẵn đối với từng user :



* Lấy tất cả địa chỉ của user :



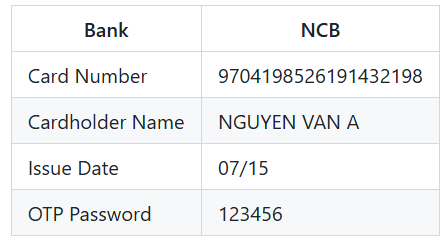
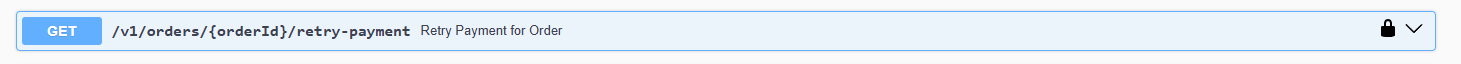
* Tiến hành gọi **Api tính phí ship** để tính phí ship đơn hàng
* Khách hàng lựa chọn phương thức thanh toán (VNPAY hoặc CASH)
* Tiến hành gọi api tạo ra một Order mới để lưu User và address vào Order (lúc này trạng thái order sẽ là PENDING), và kèm thông tin orderItem (để tạo đơn hàng), sau khi tạo xong hệ thống sẽ tự động tạo PaymentHistory với paymentStatus = PENDDING, thông tin paymentMethod là VNPAY hoặc CASH



- Khi này khách hàng lựa chọn phương thức thanh toán sẽ có 2 phương thức là tiền mặt và VNPAY, khi đó sẽ tạo PaymentHistory với paymentMethod là VNPAY hoặc CASH với paymentStatus là PENDING và liên kết với Order

* Đối với khách thanh toán qua vnpay gọi api thanh toán qua vnpay, gọi api tạo url thanh toán VNPAY và chuyển hướng

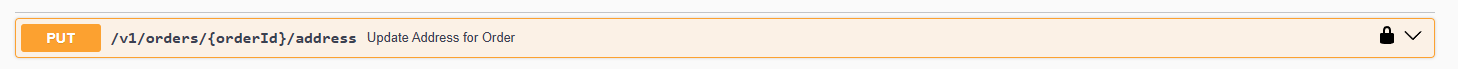


* Với thông tin thanh toán môi trường test :
* 
* Sau khi thanh toán cổng <http://localhost:5173/> (FE) sẽ nhận request từ VNPAY
* Lưu ý đối với FE sẽ nhận response từ VNPAY ở cổng <http://localhost:5173/> để biết thanh toán thành công hay thất bại (FE cần hiện thông báo đã thanh toán thành công hay thất bại)
* FE tiến hành gọi api để update trong hệ thống
* Đối với trường hợp thanh toán thành công, OrderStatus sẽ được thay đổi từ PENDING thành PROCESSING, pamentStatus sẽ chuyển thành PAID
* Đối với khách hàng thanh toán thất bại đơn hàng hệ thống sẽ cập nhập PaymentHistory thành FAILED, và đơn hàng sẽ vào phần chờ thanh toán (nếu quá 24h sẽ bị hủy đơn hàng)
* **Thanh toán lại** sẽ tạo ra PaymentHistory mới, nếu thanh toán thành công thì Order chuyển sang PROCESSING
* Đối với đơn hàng trong phần chờ thanh toán quá 24h sẽ tự động hủy đơn hàng
* Sau khi thanh toán thành công, hệ thống xử lý và cập nhật trạng thái của đơn hàng đến khi giao hàng thành công (trạng thái có thể là SHIPPED và DELIVERED sau đó)

1. **Thêm sản phẩm vào giỏ hàng rồi mới thanh toán :**

* Khách hàng có thể kiểm tra lại sản phẩm trong giỏ hàng, điều chỉnh số lượng hoặc loại bỏ sản phẩm khỏi giỏ hàng
* Khi khách hàng đã hài lòng với các sản phẩm trong giỏ hàng,
* Duyệt từng CartItem được chọn để đặt hàng, từ đó lấy được các ProductVariant thông qua response productVariantBasic của CartItemResponse
* Sau khi có được thông tin ProductVariant và tiến hành tạo order như với phần thanh toán ngay

**3. Update các thông tin đơn hàng khi đang ở trạng thái chờ :**

* Đối với các đơn hàng đã đặt thành công, chỉ được update địa chỉ của đơn hàng khi đơn hàng đang ở trạng thái PENDING hoặc PROCESSING
* Chỉ cho phép update địa chỉ khi order ở trạng thái PROCESSING hoặc PENDING

**4. Đánh giá sản phẩm :**

- Sau khi mua hàng thành công, khác hàng có thể đánh giá sản phẩm thông qua api



- Và có thể update lại thông qua api



**5. Trạng trại thái đơn hàng :**

- Gọi **API search Order** (dùng cho cả user hoặc admin): có thể sử dụng với mục đích tìm kiếm, filter, lọc các order có orderStatus = PEDDING,... sử dụng thuộc tính search **field~value** hoặc **field>value** hoặc **field<value**

****

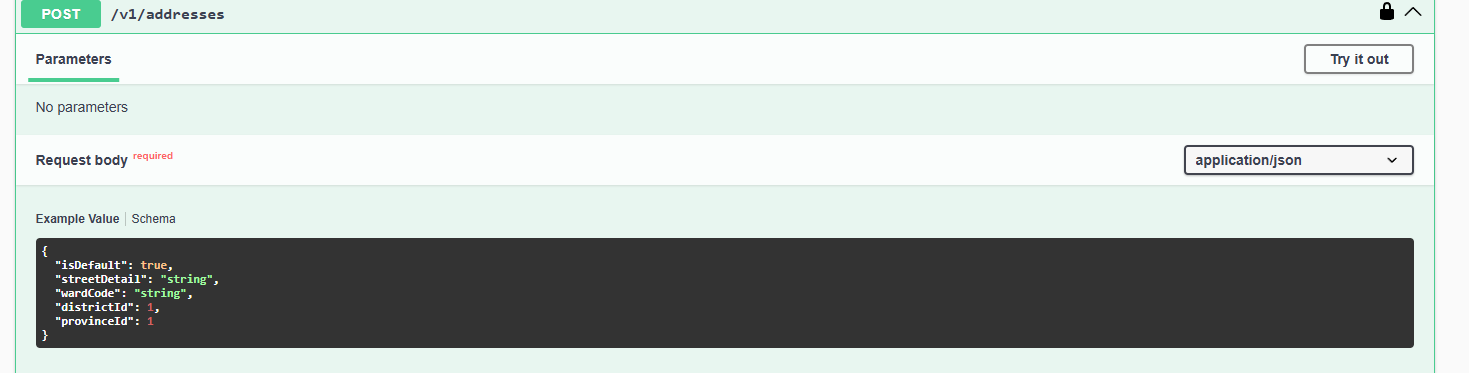
* Đối với user hệ thống sẽ tự động nhận diện và trả về các đơn hàng của chính user đó (thông qua token)
* Đối với admin hệ thống sẽ tự động lấy tất cả các đơn hàng của tất cả user

API SEARCH ORDER dùng cho nhiều mục đích, cho cả admin khi quản lý và user khi check đơn hàng của chính user

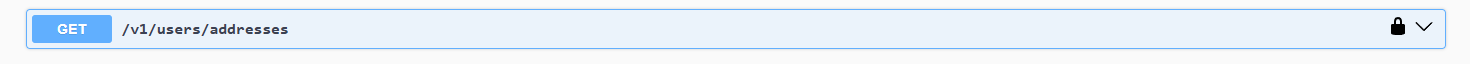
**6. Về phần address :**

* **Đối với user :**

-Thêm địa chỉ của user : cần lấy id của province và district ở các api khác



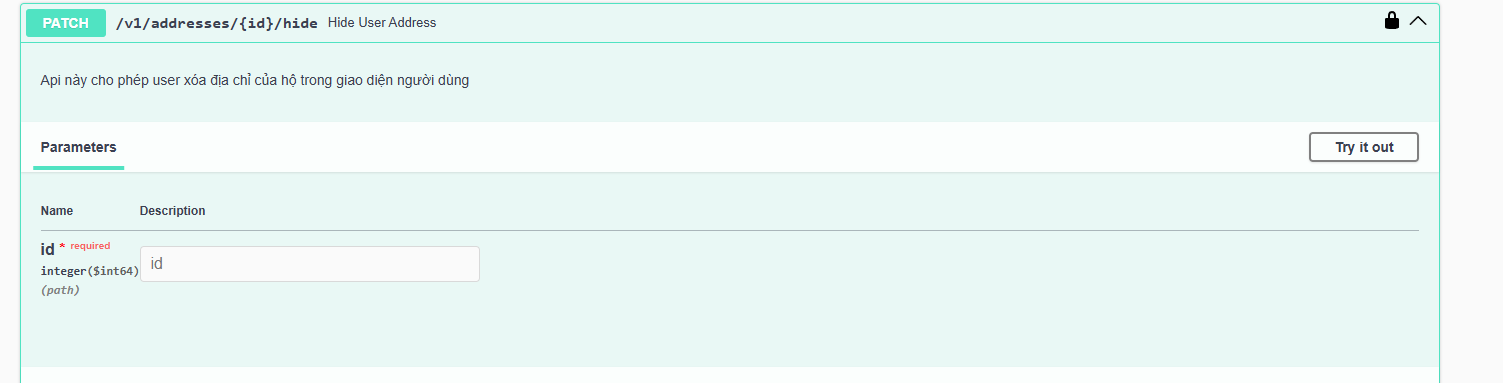
- Sau khi thêm địa chỉ cho user thành công, để lấy ra các địa chỉ user đã tạo gọi api sau để lấy tất cả địa chỉ của user :



- Khi muốn update address của user, sau khi lấy được id address của user ở trên, ta tiến hành gọi api để update address :



- Khi user muốn xóa địa chỉ trong giao diện người dùng (giao diện của riêng user), nó chỉ ẩn trong giao diện user (chuyển isVisible từ true => false) cần gọi api sau:



* **Đối với Admin :**

- Lấy tất cả địa chỉ của hệ thống (địa chỉ của nhiều user), dùng cho mục đích quản lý địa chỉ :



- Xóa mềm address (chỉ admin được quyền) (chuyển status ACTIVE => INACTIVE):



- Khôi phục address, sau khi bị xóa mềm :



**7. Quản lý doanh thu :**

- Lấy doanh thu từng tháng ( theo khoảng thời gian startDate (yyyy-MM-dd) và endDate (yyyy-MM-dd) ) :



- Lấy thông tin đơn hàng được đặt trong khoảng thời gian startDate (yyyy-MM-dd) và endDate (yyyy-MM-dd) :



r